Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

LDformer: a parallel neural network model for long-term power forecasting*

Ran TIAN‡, Xinmei LI, Zhongyu MA, Yanxing LIU, Jingxia WANG, Chu WANG

College of Computer Science & Engineering, Northwest Normal University, Lanzhou 730070, China

E-mail: tianran@nwnu.edu.cn; 2020211978@nwnu.edu.cn; mazybg@nwnu.edu.cn; lyanxing@nwnu.edu.cn; 2020222004@nwnu.edu.cn; 2020221992@nwnu.edu.cn

Received Nov. 3, 2022; Revision accepted Feb. 13, 2023; Crosschecked Sept. 5, 2023

Abstract: Accurate long-term power forecasting is important in the decision-making operation of the power grid and power consumption management of customers to ensure the power system's reliable power supply and the grid economy's reliable operation. However, most time-series forecasting models do not perform well in dealing with long-time-series prediction tasks with a large amount of data. To address this challenge, we propose a parallel time-series prediction model called LDformer. First, we combine Informer with long short-term memory (LSTM) to obtain deep representation abilities in the time series. Then, we propose a parallel encoder module to improve the robustness of the model and combine convolutional layers with an attention mechanism to avoid value redundancy in the attention mechanism. Finally, we propose a probabilistic sparse (ProbSparse) self-attention mechanism combined with UniDrop to reduce the computational overhead and mitigate the risk of losing some key connections in the sequence. Experimental results on five datasets show that LDformer outperforms the state-of-the-art methods for most of the cases when handling the different long-time-series prediction tasks.

Key words: Long-term power forecasting; Long short-term memory (LSTM); UniDrop; Self-attention mechanism
https://doi.org/10.1631/FITEE.2200540 **CLC number:** TP183

1 Introduction

Power forecasting is an important part of power system planning and the basis of the economic operation of power systems. Accurate power forecasting helps provide a reliable decision-making basis for power system planning and operation, thus reducing power production costs (Ciechulski and Osowski, 2021). Traditional prediction methods have achieved great results in meteorology, finance, industry, and transportation for short-term power forecasting problems. However, existing time-series methods still have limitations in long-time-series prediction:

1. Due to the highly complex and large-scale long-time-series data, traditional methods are limited in efficiently handling high-dimensional large data and representing complex functions (Han et al., 2021). It will forget some data and ignore the long-term dependency of the time-series data.

2. The model structure of traditional methods is not stable when long-term power forecasting is performed.

3. The existing attention mechanism is prone to losing key connections between sequences when

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No.71961028), the Key Research and Development Program of Gansu Province, China (No.22YF7GA171), the University Industry Support Program of Gansu Province, China (No.2023QB-115), the Innovation Fund for Science and Technology-based Small and Medium Enterprises of Gansu Province, China (No.23CXGA0136), and the Scientific Research Project of the Lanzhou Science and Technology Program, China (No.2018-01-58)

ORCID: Ran TIAN, https://orcid.org/0000-0003-4435-580X; Xinmei LI, https://orcid.org/0000-0002-5595-9230; Zhongyu MA, https://orcid.org/0000-0001-8809-0685; Yanxing LIU, https://orcid.org/0000-0002-0554-3683; Jingxia WANG, https://orcid.org/0000-0003-0696-9982; Chu WANG, https://orcid.org/0000-0002-8687-9911

capturing the dependencies among long-time-series data, leading to the degradation of prediction performance.

Aiming at the problem of long-time-series prediction, we propose LDformer, a parallel neural network model for long-term power forecasting based on the Informer framework (Zhou et al., 2021). The contributions of this study are summarized as follows:

1. We propose a parallel neural network model called LDformer, which combines the advantages of Informer and long short-term memory (LSTM) to effectively solve the problem of long-term power forecasting. Using LSTM to learn the long-term correlation of time-series data, the model has good long-term forecasting performance.

2. We propose a parallel encoder module that combines a convolutional layer and an attention mechanism to avoid value redundancy in the attention mechanism. Several experiments on different datasets validate the effectiveness and robustness of the parallel encoder and the convolutional layer.

3. We propose a probabilistic sparse (ProbSparse) self-attention mechanism combined with UniDrop. UniDrop does not need additional computational overhead or external resources. Combining UniDrop, ProbSparse attention mechanism can mitigate the risk of losing some key connections in the sequence.

2 Related works

Long-time-series prediction plays an essential role in decision-making in many fields, such as economy, transportation, medicine, hydrology, and energy (Zhang D et al., 2018; Chakraborty et al., 2019; Chuku et al., 2019; Xie and Lou, 2019; Marcjasz et al., 2020). However, the prediction model may produce inaccurate results due to different patterns of the actual time series. In this paper, we analyze both machine learning and deep learning.

2.1 Machine learning

In time-series prediction, the Autoregressive Integrated Moving Average (ARIMA) model is widely used in various fields. Chen JF et al. (1995) developed an adaptive ARIMA model for short-term power forecasting of a power generation system. Viccione et al. (2020) applied the ARIMA model for tank water level prediction and analysis. Many researchers have also achieved better results by improving ARIMA models or combining ARIMA with other models. Xu et al. (2017) proposed an ARIMA and Kalman filter-based approach applied to road traffic real-time state prediction. Khan et al. (2020) proposed a wavelet-based hybrid ARIMA model. However, although the ARIMA model has achieved great success in stable time-series prediction, there are no almost pure stationary data in the real-time-series data. Therefore, the application of the ARIMA model is limited by the data characteristics and is less general. As a result, many models other than ARIMA models have been applied to time-series forecasting. Syriopoulos et al. (2021) used a support vector machine (SVM) to predict shipping prices. Based on SVM, Ding et al. (2021) developed a time-series model based on the least squares support vector machine (LSSVM) and achieved better results. Nóbrega and Oliveira (2019) proposed sequential learning methods for the Kalman filter and a limited learning machine for regression and time-series prediction, obtaining better results. In addition to the normal case, many researchers have also modeled predictions for time-series data with outliers. Chen XY and Sun (2022) proposed a Bayesian time-factor decomposition framework for modeling and predicting multidimensional time series of specific spatiotemporal data in the presence of missing values. However, machine learning methods cannot obtain more accurate results for complex prediction problems.

2.2 Deep learning

With the development of deep learning, researchers have found that deep learning applies to complex time-series problems (Kim et al., 2018). Many scholars use convolutional neural networks (CNNs) to model and predict time-series data (Guo et al., 2019; Hosseini and Talebpour, 2019). Cao and Wang (2019) applied CNN to stock forecasting. However, a CNN is more suitable for spatial correlation than time series. As a result, recurrent neural networks (RNNs) have emerged. Hu et al. (2020) applied RNNs to traffic flow prediction based on time-series analysis. Min et al. (2019) proposed an RNN-based intent inference model to solve the time-series prediction problem. Many studies have proven that RNNs have obvious advantages in time-series prediction (Xiao et al., 2019; Zhang L et al., 2019). However, when the time-series is too

long, the problems of gradient disappearance and gradient explosion may occur in RNN training (Zheng and Huang, 2020).

Karevan and Suykens (2020) proposed LSTM to solve these problems. LSTM is more suitable for long-time-series than RNN. Therefore, many deep learning models based on LSTM are applied to time-series prediction. Zhang TJ et al. (2019) used LSTM for gas concentration prediction. Miao et al. (2020) proposed a novel LSTM framework for short-term fog prediction, consisting of an LSTM and a fully connected layer. Their experiments showed that the framework outperformed K-nearest neighbor (KNN), AdaBoost, and CNN algorithms. Gai et al. (2021) proposed a new parking space prediction model based on LSTM, providing a more accurate prediction. Many researchers have also proposed a combined model of LSTM and other models (Ran et al., 2019; Wang ZP et al., 2020), which proves the feasibility of LSTM in time-series prediction. However, we can learn from Khandelwal et al. (2018) that the adequate context size of the language model using LSTM is approximately 200 tokens on average. Nevertheless, the model can only distinguish 50 tokens in the vicinity, which indicates that even LSTM has difficulty capturing long-term dependencies.

In recent years, the transformer has been applied to many fields to perform long-time-series prediction tasks. Wu N et al. (2020) applied the transformer model to the prediction of influenza-like diseases, and this method can learn complex patterns and dynamics from time-series data using the self-attention mechanism. As a general framework, the transformer can not only be applied to univariate and multivariate time-series data, but can also be used for time-series embedding. However, because the point-by-point dot product in the transformer architecture is insensitive to the local context, its spatial complexity is too large. Therefore, Li et al. (2019) proposed the LogSparse transformer, which improves the prediction accuracy of time series with fine granularity and strong long-term dependence under a limited memory budget. Zhou et al. (2021) proposed the Informer and applied it to electricity consumption planning. Unlike the self-attention mechanism used in the transformer (Vaswani et al., 2017), this model proposes a new ProbSparse self-attention (Zhou et al., 2021), which minimizes the complexity and improves the transformer model's calculation, memory, and efficiency. Although the existing time-series prediction methods have promoted the development of the time-series field to a certain extent, there is still room for improvement. Because of the increase in data volume, many time-series prediction models overlook deep representation abilities of temporal data, and also have problems such as numerous parameters, large memory consumption, and long operation time.

Compared with previous work, LDformer not only considers the deep representation of temporal data, but also mitigates the risk of losing critical connections between sequences while ensuring minimum complexity. In addition, LDformer combines the convolutional layer and the attention mechanism to obtain a parallel encoder module, which prevents redundancy in the attention mechanism while improving model robustness. These methods effectively improve the prediction accuracy.

3 Preliminaries

As shown in Eq. (1), the objective of long-term power forecasting is to use model f to predict the power load data of a certain data point at n time steps based on historical power load data from multiple data points at m time steps.

$$[\boldsymbol{X}^{t-m+1}, \boldsymbol{X}^{t-m+2}, \cdots, \boldsymbol{X}^{t}] \xrightarrow{f(\cdot)} [y^{t+1}, y^{t+2}, \cdots, y^{t+n}], \tag{1}$$

where $\boldsymbol{X}^t=\{\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, \cdots, \boldsymbol{x}_{L_{\text{input}}}^t\}$ is the set of features of the input data point selected by the prediction task at time t and L_{input} is the number of input data points selected by the prediction task. $\boldsymbol{x}=\{x_1, x_2, \cdots, x_k\}$, where k is the number of features on the input data point. y^{t+1} represents the predicted value of the prediction task's data point at time t+1.

4 Long-time-series prediction model

The overall framework of LDformer is as shown in Fig. 1. LDformer contains LSTM, encoder, and decoder structures. First, the time-series data enter the

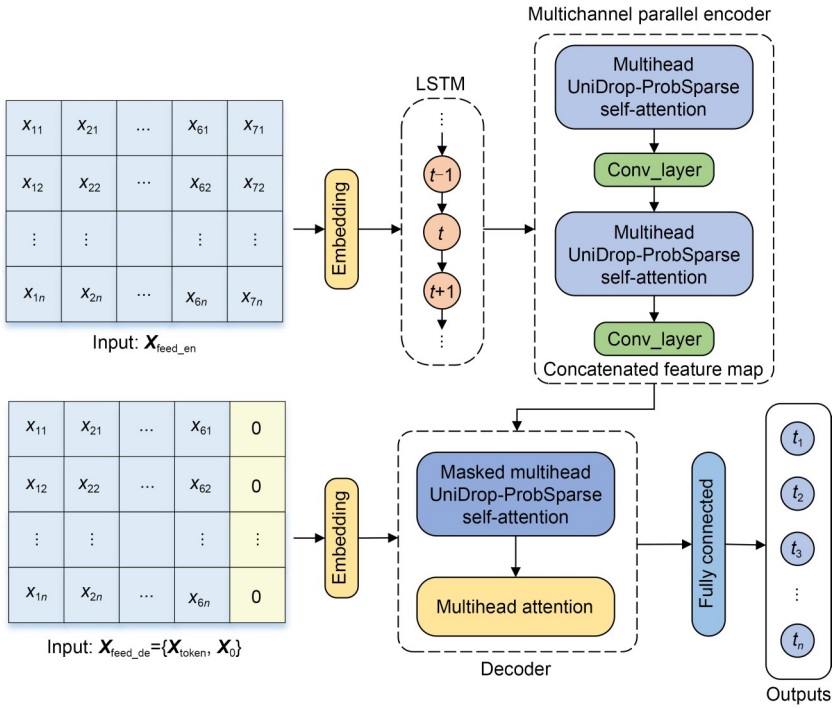

Fig. 1 Framework of LDformer

embedding layer, at which data, location, and time information is converted into a uniform dimension and then merged. The processed data then enter the LSTM, allowing for more effective use of long-term temporal information. Second, the data enter the encoder, which uses a multichannel parallel mode to improve model robustness to receive long-sequence inputs. At the same time, we extract the main features by adding convolution layers in the encoder module, and the convolution layer can generate a concentrated attention feature map to reduce feature redundancy. The decoder receives long-sequence inputs and predicts the output element immediately in the form of generation. After the embedding layer, the data enter the decoder. To ensure that the output decoded at time t depends only on the output before time t, $\boldsymbol{X}_0$ is a placeholder for the target sequence, padded with 0. A mask is added to the first attention of the decoder to prevent the target information from being used earlier. Finally, the prediction results of the last column are outputted through a fully connected layer.

4.1 Embedding layers with multiple perspectives

First, the preprocessed data are unified into the same dimension through data encoding, position encoding, and timestamp encoding, and then the final embedding result is obtained by summing the results of these three encodings.

1. Data embedding (DE): Convert data dimension x_{in} to uniform dimension d_{model} using one-dimensional (1D) convolution. The formula is as follows:

$$\text{DE}=\text{conv1D}\left(x_{\text{in}}, d_{\text{model}}\right). \tag{2}$$

2. Position embedding (PE) (Vaswani et al., 2017): Compared to RNNs, positional embedding uses a different approach where elements within the input sequence are processed concurrently, thereby preserving the positional information of each element within the sequence. Although the processing speed of RNN is higher than that of PE, it ignores the order of elements in the sequence, so we choose positional embedding. The formulae are as follows:

$$\text{PE}\left(\text{pos}, 2i\right)=\sin\left(\text{pos}/10\,000^{\frac{2i}{d_{\text{model}}}}\right), \tag{3}$$

$$\text{PE}\left(\text{pos}, 2i+1\right)=\cos\left(\text{pos}/10\,000^{\frac{2i}{d_{\text{model}}}}\right), \tag{4}$$

where pos denotes the position of the element in the sequence, d_{model} denotes the dimension of the element vector, and i denotes the position of the element vector.

3. Time embedding: There are various methods for time embedding: month_embed, day_embed, weekday_embed, hour_embed, and minute_embed. The time slice of the dataset used in this paper is mainly in hours and minutes, so hour_embed and minute_embed are chosen to obtain the timestamp encoding results.

The sum of these three embedding components is the output of the final embedding layer. The overall embedding layer structure is shown in Fig. 2.

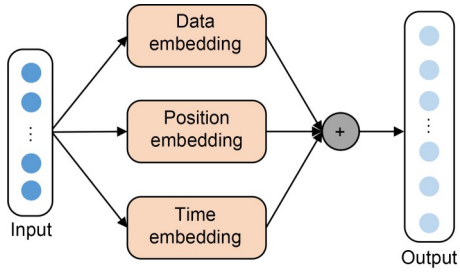

Fig. 2 Embedding layer structure

The data outputted by embedding are sent to LSTM for feature extraction. For not affecting the input of the subsequent encoder, this study makes the output of LSTM consistent with the output of embedding; i.e., the LSTM output dimension remains as d_{model}. The output of LSTM is the input of the encoder. LSTM can perform better in long sequences. Therefore, using LSTM can extract deep representation abilities in the time series and improve the prediction accuracy.

4.2 Multichannel parallel encoder module

We build the encoder module by combining the attention mechanism with the convolutional layer. It takes four channels of length L, $L/2$, $L/4$, and $L/8$, and executes them in parallel. The convolutional layer performs dimensional pruning and reduces the memory footprint before the output of the upper layer is sent to the lower layer of the multihead attention module. The convolutional layer has one fewer layer than the encoder. The multichannel parallel encoder module is shown in Fig. 3.

The encoder is used mainly to extract robust remote dependencies from time-series data. The overall architecture of the encoder is roughly the same as that of the transformer. It includes mainly two sublayers, the multihead attention layer (ProbSparse self-attention mechanism combined with UniDrop) and the feed-forward layer composed of two linear mappings. A batch normalization layer follows both sublayers, with jump connections between the sublayers.

4.2.1 ProbSparse self-attention mechanism combined with UniDrop

ProbSparse self-attention mechanism may create some risk of critical connection loss. To address this

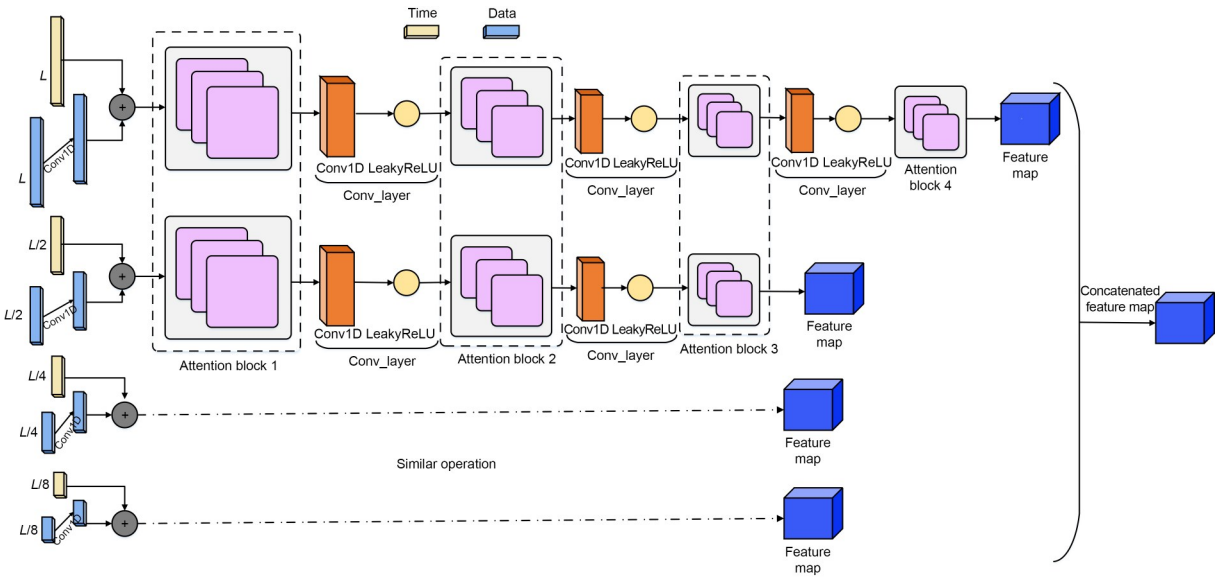

Fig. 3 Multichannel parallel encoder module (References to color refer to the online version of this figure)

problem, we propose a ProbSparse self-attention mechanism combined with UniDrop. Unlike the self-attention mechanism and ProbSparse self-attention mechanism, we consider other possible problems while retaining their advantages.

The canonical self-attention mechanism consists of a query and a set of key-value pairs. The formula is as follows (Vaswani et al., 2017):

$$\boldsymbol{A}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \mathrm{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d}}\right)\boldsymbol{V}, \tag{5}$$

where $\boldsymbol{Q}\in\mathbb{R}^{L_Q\times d}$, $\boldsymbol{K}\in\mathbb{R}^{L_K\times d}$, $\boldsymbol{V}\in\mathbb{R}^{L_V\times d}$, and d is the input dimension. This method has high time complexity. Therefore, Zhou et al. (2021) proposed ProbSparse self-attention. However, due to the large number of parameters in the attention mechanism and the tendency of overfitting and loss of key connections between sequences, we introduce the UniDrop technique (Wu Z et al., 2021), in which feature dropout (FD) can randomly suppress some neurons in the network with a certain probability. FD-1 is applied to attention weights $\boldsymbol{A}$ to improve the generalization of multihead attention. FD-2 is applied after the activation function between the two linear transformations of the feed-forward network sublayer. However, it is directly used to the attention weights $\boldsymbol{A}$, where a drop value $A(i, j)$ means ignoring the relationship between tokens i and j. Thus, a larger FD-1 means a larger risk of losing some critical information from sequence positions. We add dropout to $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ to alleviate this potential risk before calculating attention. FD-4 is used for the output features before the linear transformation. The overall structure is shown in Fig. 4.

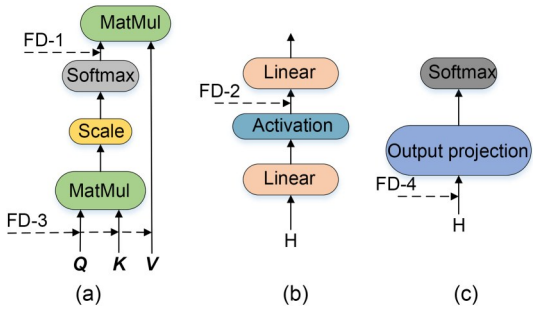

Fig. 4 UniDrop structure: (a) attention; (b) feed forward; (c) output prediction (MatMul: matrix multiplication; H: hidden layer)

Define the attention of the i^{th} row, $\boldsymbol{q}_i'$ of $\boldsymbol{Q}'$, $\boldsymbol{K}'$, $\boldsymbol{V}'$ obtained after dropout as a kernel smoother in the form of probability:

$$\boldsymbol{A}(\boldsymbol{q}_i',\boldsymbol{K}',\boldsymbol{V}') = \sum_j \frac{k(\boldsymbol{q}_i',\boldsymbol{k}_j')}{\sum_l k(\boldsymbol{q}_i',\boldsymbol{k}_l')}\boldsymbol{v}_j' = E_{p(\boldsymbol{k}_j'|\boldsymbol{q}_i')}[\boldsymbol{v}_j']. \tag{6}$$

In Eq. (6), the attention of the i^{th} query on all keys is defined as probability $p(\boldsymbol{k}_j'|\boldsymbol{q}_i')$, whose formula is

$$p(\boldsymbol{k}_j'|\boldsymbol{q}_i') = \frac{k(\boldsymbol{q}_i',\boldsymbol{k}_j')}{\sum_l k(\boldsymbol{q}_i',\boldsymbol{k}_l')}, \tag{7}$$

where $p(\boldsymbol{k}_j'|\boldsymbol{q}_i')$ and $k(\boldsymbol{q}_i',\boldsymbol{k}_j')$ select the asymmetric exponential kernel $\exp(\boldsymbol{q}_i'(\boldsymbol{k}_j')^{\mathrm{T}}/\sqrt{d})$. Self-attention combines these values and the output obtained based on $p(\boldsymbol{k}_j'|\boldsymbol{q}_i')$. However, to avoid the near-uniform distribution $q(\boldsymbol{k}_j|\boldsymbol{q}_i)=1/L_{\boldsymbol{k}}$ of $p(\boldsymbol{k}_j'|\boldsymbol{q}_i')$ so that the attention values become the sum of $\boldsymbol{V}'$, resulting in input redundancy, we use Kullback–Leibler (KL) dispersion to select important queries through the similarity distribution between p and q (Zhou et al., 2021). The formula is as follows:

$$\mathrm{KL}(q\|p) = \ln\sum_{l=1}^{L_{\boldsymbol{K}'}} \mathrm{e}^{\frac{\boldsymbol{q}_i'(\boldsymbol{k}_j')^{\mathrm{T}}}{\sqrt{d}}} - \frac{1}{L_{\boldsymbol{K}'}}\sum_{j=1}^{L_{\boldsymbol{K}'}}\frac{\boldsymbol{q}_i'(\boldsymbol{k}_j')^{\mathrm{T}}}{\sqrt{d}} - \ln L_{\boldsymbol{K}'}. \tag{8}$$

Dropping the constant, the sparsity measure of the i^{th} query can be defined as

$$M(\boldsymbol{q}_i',\boldsymbol{K}') = \ln\sum_{j=1}^{L_{\boldsymbol{K}'}} \mathrm{e}^{\frac{\boldsymbol{q}_i'(\boldsymbol{k}_j')^{\mathrm{T}}}{\sqrt{d}}} - \frac{1}{L_{\boldsymbol{K}'}}\sum_{j=1}^{L_{\boldsymbol{K}'}}\frac{\boldsymbol{q}_i'(\boldsymbol{k}_j')^{\mathrm{T}}}{\sqrt{d}}. \tag{9}$$

Eq. (9) traverses all queries that require the computation of each dot-product pair, but the log-sum-exp (LSE) operation may have numerical stability issues. Based on this, the above formula is improved to obtain the final sparsity measure formula:

$$M(\boldsymbol{q}_i',\boldsymbol{K}') = \max_j\left\{\frac{\boldsymbol{q}_i'(\boldsymbol{k}_j')^{\mathrm{T}}}{\sqrt{d}}\right\} - \frac{1}{L_{\boldsymbol{K}'}}\sum_{j=1}^{L_{\boldsymbol{K}'}}\frac{\boldsymbol{q}_i'(\boldsymbol{k}_j')^{\mathrm{T}}}{\sqrt{d}}. \tag{10}$$

Based on the above steps, we obtain ProbSparse self-attention combined with UniDrop by allowing each key to pay attention to only u dominant queries:

$$\boldsymbol{A}\left(\boldsymbol{Q}', \boldsymbol{K}', \boldsymbol{V}'\right) = \text{Softmax}\left(\frac{\bar{\boldsymbol{Q}}'(\boldsymbol{K}')^{\mathrm{T}}}{\sqrt{d}}\right)\boldsymbol{V}', \quad (11)$$

where $\bar{\boldsymbol{Q}}'$ is a sparse matrix of the same size as $\boldsymbol{q}'$ containing only top-u queries under the sparse metric, taking the part of $\boldsymbol{q}'$ with a higher probability. We set $u=c\ln L_{\boldsymbol{Q}'}$, where c is the sampling factor. The algorithmic procedure for ProbSparse self-attention combined with UniDrop is shown in Algorithm 1.

Algorithm 1 ProbSparse self-attention mechanism combined with UniDrop

Input: tensors $\boldsymbol{Q}\in\mathbb{R}^{m\times d}$, $\boldsymbol{K}\in\mathbb{R}^{n\times d}$, $\boldsymbol{V}\in\mathbb{R}^{n\times d}$

1 Initialization: set hyperparameter c, $u=c\ln m$, $U=m\ln n$, and dropout parameter p

2 $\boldsymbol{Q}'$=dropout($\boldsymbol{Q}$), $\boldsymbol{K}'$=dropout($\boldsymbol{K}$), $\boldsymbol{V}'$=dropout($\boldsymbol{V}$)

3 Arbitrarily select U dot-product pairs from $\boldsymbol{K}'$ as $\bar{\boldsymbol{K}}'$

4 Sample $\boldsymbol{K}'$ and set the sample score $\bar{\boldsymbol{S}}=\boldsymbol{Q}'(\bar{\boldsymbol{K}}')^{\mathrm{T}}$

5 Each $\boldsymbol{q}_i$ on K_sample calculates the M value

6 Set top-u queries as $\bar{\boldsymbol{Q}}'$ based on formula M

7 Set $\boldsymbol{A}_0=\text{Softmax}\left(\bar{\boldsymbol{Q}}'(\boldsymbol{K}')^{\mathrm{T}}/\sqrt{d}\right)\boldsymbol{V}'$

8 For the value of the score of unchecked $\boldsymbol{q}_i'$ set $\boldsymbol{A}_1=\text{mean}\left(\boldsymbol{V}'\right)$

9 Set self-attention formula $\boldsymbol{A}=\left\{\boldsymbol{A}_0, \boldsymbol{A}_1\right\}$

Output: self-attention feature map $\boldsymbol{A}$

4.2.2 Convolutional layer

As a natural consequence of the ProbSparse attention mechanism combined with UniDrop, the feature mapping of the encoder produces a redundant combination of values $\boldsymbol{V}'$. In the next layer, we use convolution to extract dominant features for processing so that they generate a focused attentional feature map, as shown in Fig. 3, which will largely reduce the temporal dimension of the input. CNN is good at identifying simple patterns in data and generating complex patterns in more advanced layers. Conv1D is effective in obtaining features of interest from data whose locations are not highly correlated, and Conv1D can be well applied to time-series analysis of sensor data. Therefore, we select Conv1D to extract features and set its convolution kernel as 3×3. The formula is as follows:

$$\boldsymbol{X}_{j+1}^{t} = \text{LeakyReLU}\left(\text{Conv1D}\left(\left[\boldsymbol{X}_{j}^{t}\right]_{\text{AB}}\right)\right), \quad (12)$$

where $[\cdot]_{\text{AB}}$ contains the basic operations in the multi-head attention and attention block, and Conv1D uses the LeakyReLU activation function to execute in the time dimension. LeakyReLU is a variant of ReLU. It introduces some changes when the input value is less than 0, alleviates the sparsity of ReLU, and inherits the advantages of ReLU. It can also speed up the convergence, alleviate the gradient disappearance and explosion problems, and simplify the calculation. The LeakyReLU activation function is as follows:

$$\text{LeakyReLU}(x)=\max(0, x)+\text{negative\_slope}\cdot\min(0, x). \quad (13)$$

4.3 Decoder

The decoder generates the time-series output by a forward process, and part of its structure can be referred to as the decoder structure in the transformer, as shown in Fig. 5.

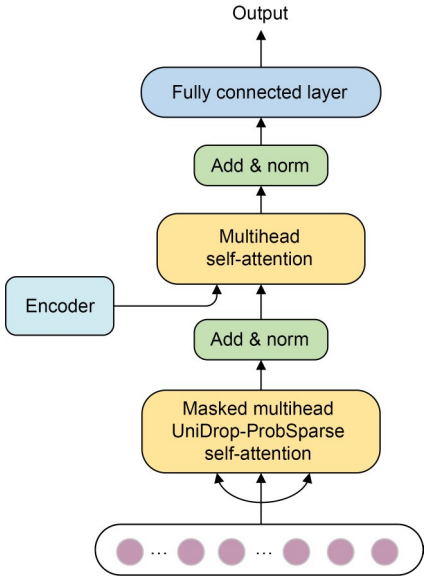

Fig. 5 Decoder for growing sequence output through forward process generation

The decoder consists of two layers of attention mechanism and a linear mapping feed-forward layer. The decoder's input vector is as follows:

$$\boldsymbol{X}_{\text{feed\_de}}^{t} = \text{Concat}\left(\boldsymbol{X}_{\text{token}}^{t}, \boldsymbol{X}_{0}^{t}\right) \in \mathbb{R}^{\left(L_{\text{token}}+L_{\text{o}}\right)\times d_{\text{model}}}, \quad (14)$$

where $\boldsymbol{X}_{\text{token}}^{t} \in \mathbb{R}^{L_{\text{token}}\times d_{\text{model}}}$ is the starting token and $\boldsymbol{X}_{0}^{t} \in \mathbb{R}^{L_{\text{o}}\times d_{\text{model}}}$ is a placeholder for the target sequence.

The first layer of attention is the ProbSparse self-attention mechanism combined with UniDrop, as shown in Eq. (11). We set the masked multihead self-attention to $-\infty$. It prevents focusing on the future position in the training process to avoid the autoregression problem. The second layer of attention is ordinary self-attention, as shown in Eq. (5). An add & norm layer follows both layers of attention. Add & norm is calculated as follows:

$$\boldsymbol{X}=\text{LayerNorm}\left(\boldsymbol{X}+\text{MultiheadAttention}\left(\boldsymbol{X}\right)\right). \quad (15)$$

Finally, the prediction results are outputted directly through a fully connected layer.

5 Experiments and analyses

5.1 Datasets

We extensively perform experiments on five datasets, namely, the ETTm1, ETTh1, ETTh2, PEMS03, and weather datasets. ETTm1, ETTh1, and ETTh2 are collectively referred to as the ETT dataset. Table 1 shows the description of the datasets.

ETT (Zhou et al., 2021): ETT is a key indicator of the long-term deployment of electricity. Here are two years of data collected from two cities in China. ETTm1 takes one data point every 15 min. ETTh1 and ETTh2 take one data point every hour. Each data point consists of the target value oil temperature (OT) and six different types of external load values, i.e., high useful load (HUFL), high useless load (HULL), middle useful load (MUFL), middle useless load (MULL), low useful load (LUFL), and low useless load (LULL). We use a multivariate prediction univariate model with six power load features to predict the target value OT. According to the time characteristics, we divide the training set, validation set, and test set by 12, 4, and 4 months, respectively.

Weather (Zhou et al., 2021): This dataset contains local climate data for nearly 1600 U.S. locations. Each data point consists of a target value wet bulb and 11 climate features. We divide the training set, validation set, and test set by 12, 4, and 4 months, respectively.

PEMS03 (Wang C et al., 2023): This dataset is the traffic flow data of the California highway network. It is divided into five-minute intervals and contains data from 307 sensors for three months from 2018/9/1 to 2018/11/30. This experiment captures three months of traffic data from the first seven sensors. The dataset is divided by 7:2:1 into a training set, test set, and validation set. The seventh sensor is used as the target sensor for the experiment.

5.2 Experimental setting

In this study, the model uses both parallel and nonparallel modes in the encoder. The parallel mode has four channels executing in parallel and three stacks. The decoder contains two stacks. The sampling factor c in the top-u formula is set to 5. The parameter in dropout is 0.1. The number of multihead attention n-heads is set to 8. When predicting the target sequence, the mean square error (MSE) is selected as the loss function, and the whole model is returned from the output of the decoder. The learning rate in the experimental setup decreases from 1e-04 and decays

Table 1 Datasets and prediction task descriptions

Dataset	Data description	Time range	Time interval	Target feature	Target length	Length of time (h)
ETTm1	Key indicators for long-term power deployment collected from two different counties in China	07/01/2016–06/26/2018	15 min	Oil temperature	{24, 36, 48, 96, 168, 336}	{6, 9, 12, 24, 42, 84}
ETTh1		07/01/2016–06/26/2018	1 h	Oil temperature	{24, 36, 48, 96, 168, 336}	{24, 36, 48, 96, 168, 336}
ETTh2		07/01/2016–06/26/2018	1 h	Oil temperature	{24, 36, 48, 96, 168, 336}	{24, 36, 48, 96, 168, 336}
Weather	Local climate data for nearly 1600 U.S. locations	01/01/2010–12/31/2013	1 h	Wet bulb	{24, 48, 96, 168, 336, 720}	{24, 48, 96, 168, 336, 720}
PEMS03	Traffic flow dataset	09/01/2018–11/30/2018	5 min	313 512	{24, 48, 96, 228, 336, 720}	{2, 4, 8, 24, 28, 60}

Target length means number of time interval. Length of time=time interval×target length

by a factor of 2 for each period. The model employs the Adam optimizer as its optimization algorithm, with 10 epochs and a batch size of 64. Moreover, prior to conducting the experiment, each input dataset is subjected to a standardization procedure.

Assessment metrics: three evaluation indexes, MSE, mean absolute error (MAE), and root mean square error (RMSE), are used. The primary reason for choosing RMSE is that, compared to MSE and other evaluation metrics, RMSE has advantages such as greater intuitiveness, stronger robustness, and easier interpretability in the assessment of model performance. Consequently, RMSE is widely adopted in numerous practical applications. For example, when RMSE is equal to 10, the regression effect can be considered to differ by 10 on average compared to the true value. The indicator equations are as follows:

$$\mathrm{MSE}=\frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{y}-\hat{\boldsymbol{y}}\right)^{2}, \tag{16}$$

$$\mathrm{MAE}=\frac{1}{n}\sum_{i=1}^{n}\left|\boldsymbol{y}-\hat{\boldsymbol{y}}\right|, \tag{17}$$

$$\mathrm{RMSE}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{y}-\hat{\boldsymbol{y}}\right)^{2}}, \tag{18}$$

where $\boldsymbol{y}$ is the true value and $\hat{\boldsymbol{y}}$ is the prediction value.

5.3 Experimental results and analyses

We conduct comparative experiments on five datasets and multiple prediction tasks, comparing widely used RNN, LSTM, and gated recurrent unit (GRU) commonly employed in long-time-series prediction, as well as the representative Informer algorithm and its Informer (np) model under nonparallel conditions, with our parallel model LDformer and nonparallel LDformer (np). The specific description of the datasets is shown in Table 1. The corresponding experimental results are shown in Tables 2–5.

Table 2 Performance comparison of short-time-series prediction tasks in the ETT dataset at different prediction lengths (24, 36, 48)

Dataset	Model	MSE			MAE			RMSE		
		24	36	48	24	36	48	24	36	48
ETTm1	RNN	5.0206	2.7855	5.7508	1.9853	1.4088	2.1783	2.2407	1.6689	2.3981
	LSTM	2.6396	2.7410	2.7252	1.3441	1.3751	1.3704	1.6247	1.6555	1.6508
	GRU	1.9783	5.0422	3.0284	1.1282	1.9861	1.4052	1.4065	2.1454	1.7402
	Informer (np)	0.0904	0.0816	0.0739*	0.2279	0.2247	0.2117*	0.3006	0.2852	0.2719*
	Informer	**0.0432**	**0.0674**	0.1055	**0.1606**	**0.2021**	0.2571	**0.2079**	**0.2597**	0.3248
	LDformer (np)	0.0605*	0.0800	0.0999	0.1962*	0.2271	0.2574	0.2458*	0.2830	0.3161
	LDformer	0.0789	0.0754*	**0.0665**	0.2301	0.2136*	**0.1993**	0.2810	0.2747*	**0.2578**
ETTh1	RNN	3.6319	4.3593	4.6225	1.6149	1.8123	1.8864	1.9057	2.0879	2.1500
	LSTM	2.1395	2.7266	2.5198	1.1835	1.3697	1.3068	1.4627	1.6512	1.5874
	GRU	0.9840	0.9556	1.9024	0.7832	0.7625	1.1236	0.9920	0.9775	1.3792
	Informer (np)	0.3420*	0.3861*	0.5013	0.5325*	0.5532*	0.6441	0.5848*	0.6314*	0.7080
	Informer	0.4798	0.5079	0.3964*	0.6475	0.6685	**0.5406**	0.6926	0.7127	0.6296*
	LDformer (np)	0.4193	**0.3458**	0.5183	0.5937	**0.5314**	0.6545	0.6475	**0.5880**	0.7199
	LDformer	**0.2492**	0.4304	**0.3782**	**0.4361**	0.5859	0.5579*	**0.4992**	0.6560	**0.6150**
ETTh2	RNN	2.2385	2.0350	1.4774	1.2577	1.1824	1.0111	1.4961	1.4265	1.2155
	LSTM	1.0688	1.1154	**0.6622**	0.8114	0.8529	**0.6543**	1.0338	1.0561	**0.8138**
	GRU	0.7586*	0.7955	1.0504	0.7136*	0.7204*	0.8336	0.8710*	0.8919	1.0249
	Informer (np)	1.0555	1.2231	1.7355	0.9186	1.0052	1.2214	1.0273	1.1059	1.3173
	Informer	1.2559	1.1121	2.0611	1.0338	0.9529	1.3422	1.1206	1.0545	1.4356
	LDformer (np)	0.7810	**0.5988**	0.7493	0.7721	**0.6642**	0.9541	0.8837	**0.7738**	1.0720
	LDformer	**0.5713**	0.7553*	0.7224*	**0.6384**	0.7575	0.7296*	**0.7558**	0.8691*	0.8499*

The best results are in bold, and * denotes the second-best results. MSE: mean square error; MAE: mean absolute error; RMSE: root mean square error

Table 3 Performance comparison of long-time-series prediction tasks in the ETT dataset at different prediction lengths (96, 168, 336)

Dataset	Model	MSE			MAE			RMSE		
		96	168	336	96	168	336	96	168	336
ETTm1	RNN	3.7047	4.9325	3.4728	1.6129	1.9187	1.5242	1.9247	2.2209	1.8635
	LSTM	2.7349	2.7401	2.7471	1.3728	1.3749	1.3763	1.6537	1.6553	1.6574
	GRU	3.6131	2.8525	3.1644	1.5948	1.4074	1.4829	1.9008	1.6889	1.7788
	Informer (np)	0.3387*	0.6584	1.0142	0.5361	0.7109	0.9140	0.5820*	0.8114	1.0070
	Informer	0.5603	0.6100	1.0266	0.6874	0.7097	0.9412	0.7485	0.7810	1.0132
	LDformer (np)	**0.3248**	0.5146*	**0.5357**	**0.5142**	0.6573*	**0.6519**	**0.5699**	0.7173*	**0.7319**
	LDformer	0.3523	**0.5096**	0.6346*	0.5277*	**0.6166**	0.7103*	0.5936	**0.7138**	0.7966*
ETTh1	RNN	3.5576	3.4802	2.8218	1.5451	1.5640	1.3619	1.8861	1.8655	1.6798
	LSTM	2.2462	2.2827	2.0860	1.2268	1.2244	1.1430	1.4987	1.5108	1.4443
	GRU	2.6474	1.8953	2.3861	1.3466	1.0810	1.2097	1.6271	1.3767	1.5447
	Informer (np)	**0.2577**	0.3024*	0.7343	**0.4236**	0.4721*	0.7743	**0.5076**	0.5499*	0.8569
	Informer	0.3014*	**0.2656**	0.9416	0.4639*	**0.4413**	0.8984	0.5490*	**0.5154**	0.9703
	LDformer (np)	0.4545	0.4046	0.5057*	0.6069	0.5617	**0.6421**	0.6741	0.5615	0.7111*
	LDformer	0.3106	0.4886	**0.5034**	0.4880	0.6322	0.6455*	0.5569	0.5224	**0.7095**
ETTh2	RNN	1.5415	2.0284	2.1122	1.0042	1.1726	1.1998	1.2416	1.4242	1.4533
	LSTM	**1.0823**	1.1039*	**1.1152**	**0.8412**	0.8520*	0.8568*	**1.0403**	1.0507*	**1.0560**
	GRU	1.1424*	**0.8537**	1.1582*	0.8764*	**0.7320**	**0.8390**	1.0688*	**0.9239**	1.0762*
	Informer (np)	2.2963	2.3733	2.2128	1.4216	1.4409	1.3732	1.5153	1.5405	1.4875
	Informer	2.7244	2.9010	2.1025	1.5558	1.6124	1.3379	1.6506	1.7032	1.4500
	LDformer (np)	2.1959	2.2004	2.8714	1.3798	1.3807	1.5847	1.4818	1.4834	1.6945
	LDformer	1.8590	2.5229	2.6768	1.2490	1.4819	1.5261	1.3634	1.5883	1.6361

The best results are in bold, and * denotes the second-best results. MSE: mean square error; MAE: mean absolute error; RMSE: root mean square error

Tables 2 and 3 show the prediction performance of the baseline models and our models for short- and long-time-series of the ETT dataset, respectively. As the prediction length increases, the prediction performance of LDformer continues to increase in all datasets. This demonstrates the success of LDformer in improving prediction performance in solving long-time-series prediction problems. In the ETTm1 dataset, when the prediction length is 48, MSE, MAE, and RMSE of the LDformer model proposed in this paper are 37.0%, 22.5%, and 20.6% lower than those of Informer, respectively. As the prediction length increases, the advantages of the LDformer become more apparent. When the prediction length is 168, MSE, MAE, and RMSE of LDformer are reduced by 16.5%, 13.1%, and 8.6% compared to those of Informer, respectively. When the prediction length is 336 in the ETTm1 dataset, the LDformer (np) model has the optimal result, while the LDformer model shows the suboptimal result.

When the prediction length in ETTh1 is 24, MSE, MAE, and RMSE of LDformer are 74.7%, 44.3%, and 49.7% lower than those of the traditional GRU model, respectively. After increasing the prediction length, Informer shows good performance for the ETTh1 dataset when the prediction length is 96 and 168. However, the performance of LDformer is better than those of traditional models in most cases. In the ETTh2 dataset of long-time-series prediction, the model in this paper outperforms traditional models. This phenomenon may be caused by the anisotropy of the feature dimensions. It is beyond the scope of this paper, and we will explore it in the future work. ETTh2 contains a large amount of continuous null data.

Tables 4 and 5 show the prediction performance of the baseline models and the models proposed in this paper for short- and long-time-series in the weather and PEMS03 datasets, respectively. For the weather dataset, when the prediction length is 168,

Table 4 Performance comparison of short-time-series prediction tasks in the weather and PEMS03 datasets at different prediction lengths (24, 48, 96)

Dataset	Model	MSE			MAE			RMSE		
		24	48	96	24	48	96	24	48	96
Weather	RNN	0.9135	1.0542	1.5387	0.5446	0.6193	0.8906	0.9558	1.0267	1.2404
	LSTM	0.6073	0.6203	1.0249	0.4037	0.5168	0.5648*	0.7793	0.7875	1.0123
	GRU	0.4992	0.5184	0.7345	0.3378	0.5661	0.6472	0.7066	0.7200	0.7965
	Informer (np)	0.1666	0.3414*	0.7039	0.2862	0.4285*	0.6236	0.4082	0.5843*	0.8390
	Informer	0.1594*	0.3802	0.6459	0.2809*	0.4552	0.5981	0.3993*	0.6166	0.8037
	LDformer (np)	0.1650	**0.3393**	**0.5638**	0.2905	**0.4279**	**0.5594**	0.4062	**0.5825**	**0.7508**
	LDformer	**0.1574**	0.3677	0.5653*	**0.2918**	0.4484	0.5672	**0.3968**	0.6064	0.7519*
PEMS03	RNN	0.4110	0.1498	0.8322	0.5089	0.3166	0.7091	0.6411	0.3871	0.9122
	LSTM	0.0776	0.0835*	0.2132	0.2191	0.2291	0.2644	0.2786	0.3089	0.4364
	GRU	0.0779	0.0842	0.2098	0.2206	0.2309	0.3635	0.2791	0.3002	0.4313
	Informer (np)	0.0582	0.0970	**0.1145**	0.1772	0.2158	**0.2377**	0.2413	0.3115	**0.3384**
	Informer	0.0553*	0.0891	0.1195*	0.1705*	0.2070	0.2426*	0.2352*	0.2986*	0.3457*
	LDformer (np)	**0.0551**	0.0915	0.1507	**0.1688**	**0.2054**	0.2568	**0.2348**	0.3024	0.3882
	LDformer	0.0650	**0.0865**	0.1279	0.1788	0.2055*	0.2400	0.2550	**0.2942**	0.3576

The best results are in bold, and * denotes the second-best results. MSE: mean square error; MAE: mean absolute error; RMSE: root mean square error

Table 5 Performance comparison of long-time-series prediction tasks in the weather dataset at prediction lengths of 168, 336, and 720 and the PEMS03 dataset at prediction lengths of 288, 336, and 720

Dataset	Model	MSE			MAE			RMSE		
		168	336	720	168	336	720	168	336	720
Weather	RNN	1.6785	1.9695	1.9957	1.0261	1.1659	1.1651	1.2955	1.4034	1.4126
	LSTM	0.9975	1.0505	1.0395	0.6133*	0.7713	0.7682	0.9987	1.0249	1.0195
	GRU	0.7594	0.8735	1.6688	0.6360	0.6890	0.9437	0.8714	0.9346	1.2918
	Informer (np)	0.8772	0.8803	0.8857	0.7056	0.7128	0.7244	0.9366	0.9382	0.9411
	Informer	0.7835	0.8791	0.8743	0.6681	0.7196	0.7194	0.8851	0.9376	0.9350
	LDformer (np)	0.6598*	**0.7277**	**0.7751**	0.6134	0.6485*	**0.6699**	0.8122*	**0.8531**	**0.8804**
	LDformer	**0.6194**	0.7452*	0.7881*	**0.6040**	**0.6477**	0.6807*	**0.7870**	0.8632*	0.8877*
Dataset	Model	MSE			MAE			RMSE		
		288	336	720	288	336	720	288	336	720
PEMS03	RNN	1.1370	1.0338	1.3234	0.7934	0.7569	0.8594	1.0663	1.0167	1.1504
	LSTM	0.8874	1.0287	0.3207	0.8013	0.8595	0.4365	0.9420	1.0142	0.5663
	GRU	0.2669	0.2323	0.2395	0.4190	0.3918	0.4963	0.5085	0.4637	0.5735
	Informer (np)	0.2264	0.1917*	0.2252	0.3100	0.2941	0.3112	0.4758	0.4378*	0.4746
	Informer	0.2039	0.2090	0.2264	0.2967	0.3091	0.3200	0.4515	0.4571	0.4758
	LDformer (np)	0.1759*	0.1957	**0.2184**	0.2690*	0.2885*	**0.3046**	0.4195*	0.4424	**0.4674**
	LDformer	**0.1603**	**0.1826**	0.2251*	**0.2617**	**0.2788**	0.3087*	**0.4004**	**0.4273**	0.4744*

The best results are in bold, and * denotes the second-best results. MSE: mean square error; MAE: mean absolute error; RMSE: root mean square error

MSE, MAE, and RMSE of LDformer are reduced by 29.4%, 14.4%, and 16.0%, respectively, compared to those of Informer (np). For the PEMS03 dataset, when the prediction length is 336, MSE, MAE, and RMSE of LDformer are reduced by 12.6%, 9.8%, and 6.5%, respectively, compared to those of Informer. The experiments show that the models proposed in this paper achieve better results for the PEMS03 dataset with a time interval of 5 min and the ETTm1 dataset with a time interval of 15 min.

LDformer accurately obtains the temporal information of each feature, where LSTM learns the long-term dependency of temporal data and fully exploits deep representation abilities among time-series data. The ProbSparse attention mechanism combined with UniDrop inherits the advantages of the ProbSparse attention mechanism, while avoiding the attention mechanism losing some key connections in the sequence when considering data correlation. Meanwhile, the encoder module adopts a multichannel parallel model to improve the robustness of LDformer.

We calculate the average error between the true and prediction values for different datasets. The error is plotted as a histogram, as shown in Figs. 6 and 7.

In the short-time-series prediction (Fig. 6), LDformer has the smallest average error between the prediction and true values, and the prediction accuracy is higher than that of other models. For long-time-series prediction (Fig. 7), LDformer still achieves good results. However, the results are unsatisfactory due to the availability of null data in the ETTh2 dataset. LDformer combines the advantages of Informer and LSTM, and uses four channels in the encoder to

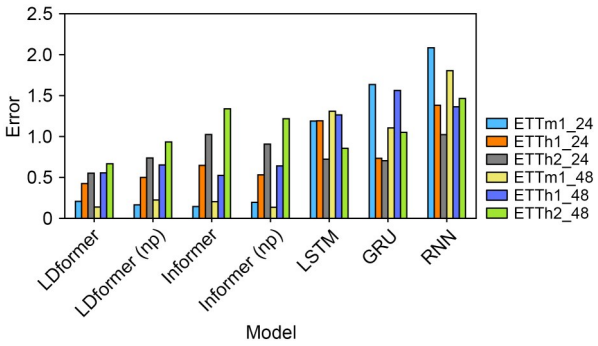

Fig. 6 Average error of the true and prediction values in the short-time-series prediction (References to color refer to the online version of this figure)

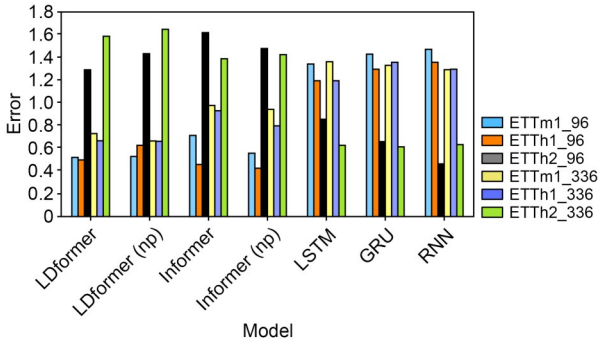

Fig. 7 Average error of the true and prediction values in the long-time-series prediction (References to color refer to the online version of this figure)

improve the stability of the model. It also uses ProbSparse self-attention combined with UniDrop to reduce the loss of some key connections in the sequence and improves the accuracy of long-time-series prediction.

We plot the average loss corresponding to different learning rates in the four models. As shown in Fig. 8, the LDformer (np) proposed in this paper converges faster than the LDformer model.

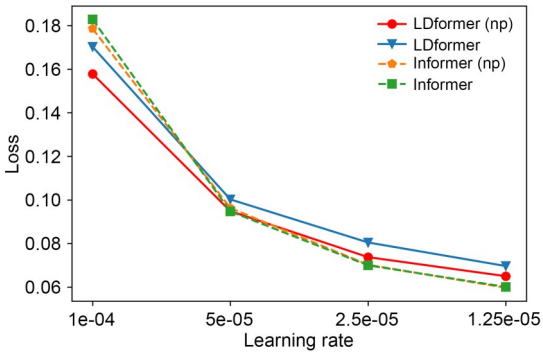

Fig. 8 Convergence of loss with a decreasing learning rate for the ETTh1 dataset at a prediction length of 24

Fig. 9 shows the training runtime profiles of several models with an increasing number of epochs. Under the early stop mechanism, both LDformer and LDformer (np) stop training at the fourth epoch, Informer stops training at the fifth epoch, and Infomer (np) stops training at the sixth epoch. Informer (np) obtains the optimal result at the sixth epoch. The training time of the models in this paper is significantly less than that of other models. Our proposed ProbSparse attention mechanism combined with UniDrop reduces the number of parameters without increasing the time complexity. Convolutional layers extract dominant

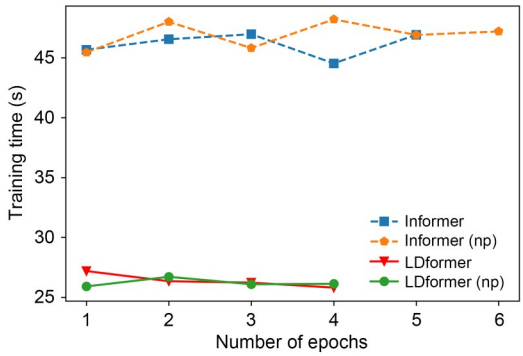

Fig. 9 Training time comparison

features, reducing the temporal feature dimension and avoiding redundancy in the attention mechanism. This is the key factor that reduces the training time.

We perform ablation experiments for each innovation point to further evaluate the effectiveness of the individual components in LDformer. Table 6 shows several models with various innovation points removed.

We compare the MSE and MAE of our models on the four prediction tasks with those of the models without innovation point. Table 7 shows that the cancellation of each innovation point affects the results. Through LSTM, LDformer can extract deep representation abilities from time-series data. This is achieved by the use of LSTM memory cells and gate mechanisms. Through LSTM memory cells and gate mechanisms, LSTM can effectively capture long-term dependencies and longer contextual information. UniDrop reduces the number of parameters and computational complexity, improving the model's parallel computing capabilities. Multiple parallel channels contribute to a more stable model structure, by accepting various long sequences, which increases the model's robustness to input noise and variations, making it more adaptable to different input conditions and data distributions. The convolutional layer reduces the potential complexity and computational cost of the attention

Table 6 Introduction to the ablation experiment modules

Model	LSTM	Parallel		Nonparallel	ProbSparse self-attention combined with UniDrop	Convolutionl layer
		2-way	4-way			
Model 0				√	√	√
Model 1			√		√	
Model 2	√			√		√
Model 3	√		√			√
Model 4	√			√	√	
Model 5	√		√		√	
Model 6	√	√			√	√

Table 7 The MSE and MAE of different models for the ETTm1 dataset at prediction lengths of 48 and 168 and the ETTh1 dataset at prediction lengths of 24 and 336

Model	MSE				MAE			
	ETTm1_48	ETTm1_168	ETTh1_24	ETTh1_336	ETTm1_48	ETTm1_168	ETTh1_24	ETTh1_336
Model 0	0.1977	0.5817	0.4452	0.6005	0.3713	0.6859	0.6051	0.7049
Model 1	0.1719	0.7255	0.3028	0.6320	0.3471	0.7808	0.4783	0.7170
Model 2	0.1224	**0.5064**	0.3211	0.9754	0.2849	0.6318	0.5132	0.9306
Model 3	0.1389	0.6188	0.2690	0.5424	0.3119	0.7034	0.4529	0.6655
Model 4	0.0838	0.5101	0.2843	0.8499	0.2304	0.6338	0.4617	0.8273
Model 5	0.1400	0.5310	0.2574	0.6655	0.3050	0.6574	**0.4291**	0.7562
Model 6	0.1254	0.6792	0.3057	0.6221	0.2916	0.7595	0.4964	0.7159
LDformer (np)	0.0999	0.5146	0.4193	0.5057	0.2574	0.6573	0.5937	**0.6421**
LDformer	**0.0665**	0.5096	**0.2492**	**0.5034**	**0.1993**	**0.6166**	0.4361	0.6455

The best results are in bold. MSE: mean square error; MAE: mean absolute error

mechanism by reducing the number of parameters. By combining these modules, LDformer achieves optimal results on most prediction tasks.

6 Conclusions

In this paper, we propose a long-term power forecasting model LDformer and its nonparallel state LDformer (np) to solve the long-time-series prediction problem with the power dataset ETT as an example. The LDformer model is useful for obtaining more accurate prediction results without increasing the time complexity. First, ProbSparse self-attention mechanism combined with UniDrop is proposed to replace ProbSparse self-attention mechanism, effectively avoiding the risk of losing key connections between sequences without increasing the complexity. Second, two perspectives, parallel and nonparallel, are used in the encoder module for comparison. Considering the number of parameters and model stability, we use convolutional layers for data extraction between attention modules. Finally, we combine the improved model with LSTM to capture the long-range time-series information and extract deep representation abilities in the time series to improve prediction accuracy. Experimental results confirm that LDformer performs better on long-time-series prediction tasks for short-interval datasets. For each innovation point proposed in this study, ablation experiments have been conducted to demonstrate the innovation points' feasibility. However, the method presented in this study still has potential for improvement in dealing with long-term-series prediction tasks for long-interval datasets.

Contributors

Ran TIAN designed the research. Xinmei LI developed the methodology, curated the data, and worked on the software. Zhongyu MA conducted the investigation. Yanxing LIU processed the data. Jingxia WANG conducted the data visualization and result validation. Chu WANG verified the experimental results. Xinmei LI drafted the paper. Ran TIAN revised and finalized the paper.

Compliance with ethics guidelines

Ran TIAN, Xinmei LI, Zhongyu MA, Yanxing LIU, Jingxia WANG, and Chu WANG declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Cao JS, Wang JH, 2019. Stock price forecasting model based on modified convolution neural network and financial time series analysis. *Int J Commun Syst*, 32(12):e3987. https://doi.org/10.1002/dac.3987

Chakraborty T, Chattopadhyay S, Ghosh I, 2019. Forecasting dengue epidemics using a hybrid methodology. *Phys A Stat Mech Appl*, 527:121266. https://doi.org/10.1016/j.physa.2019.121266

Chen JF, Wang WM, Huang CM, 1995. Analysis of an adaptive time-series autoregressive moving-average (ARMA) model for short-term load forecasting. *Electr Power Syst Res*, 34(3):187-196. https://doi.org/10.1016/0378-7796(95)00977-1

Chen XY, Sun LJ, 2022. Bayesian temporal factorization for multidimensional time series prediction. *IEEE Trans Patt Anal Mach Intell*, 44(9):4659-4673. https://doi.org/10.1109/TPAMI.2021.3066551

Chuku C, Simpasa A, Oduor J, 2019. Intelligent forecasting of economic growth for developing economies. *Int Econ*, 159:74-93. https://doi.org/10.1016/j.inteco.2019.06.001

Ciechulski T, Osowski S, 2021. High precision LSTM model for short-time load forecasting in power systems. *Energies*, 14(11):2983. https://doi.org/10.3390/en14112983

Ding M, Zhou H, Xie H, et al., 2021. A time series model based on hybrid-kernel least-squares support vector machine for short-term wind power forecasting. *ISA Trans*, 108:58-68. https://doi.org/10.1016/j.isatra.2020.09.002

Gai SX, Zeng XQ, Yuan TF, 2021. Parking volume forecast of railway station garages based on passenger behaviour analysis using the LSTM network. *J Adv Transp*, 2021:6688609. https://doi.org/10.1155/2021/6688609

Guo SN, Lin YF, Li SJ, et al., 2019. Deep spatial-temporal 3D convolutional neural networks for traffic data forecasting. *IEEE Trans Intell Transp Syst*, 20(10):3913-3926. https://doi.org/10.1109/TITS.2019.2906365

Han ZY, Zhao J, Leung H, et al., 2021. A review of deep learning models for time series prediction. *IEEE Sens J*, 21(6):7833-7848. https://doi.org/10.1109/JSEN.2019.2923982

Hosseini MK, Talebpour A, 2019. Traffic prediction using time-space diagram: a convolutional neural network approach. *Transp Res Rec J Transp Res Board*, 2673(7):425-435. https://doi.org/10.1177/0361198119841291

Hu R, Chiu YC, Hsieh CW, 2020. Crowding prediction on mass rapid transit systems using a weighted bidirectional recurrent neural network. *IET Intell Transp Syst*, 14(3): 196-203. https://doi.org/10.1049/iet-its.2018.5542

Karevan Z, Suykens JAK, 2020. Transductive LSTM for time-series prediction: an application to weather forecasting. *Neur Netw*, 125:1-9. https://doi.org/10.1016/j.neunet.2019.12.030

Khan MH, Muhammad NS, El-Shafie A, 2020. Wavelet based hybrid ANN-ARIMA models for meteorological drought forecasting. *J Hydrol*, 590:125380. https://doi.org/10.1016/j.jhydrol.2020.125380

Khandelwal U, He H, Qi P, et al., 2018. Sharp nearby, fuzzy far away: how neural language models use context. Proc 56th Annual Meeting of the Association for Computational Linguistics, p.284-294. https://doi.org/10.18653/v1/P18-1027

Kim SH, Lee G, Kwon GY, et al., 2018. Deep learning based on multi-decomposition for short-term load forecasting. *Energies*, 11(12):3433. https://doi.org/10.3390/en11123433

Li SY, Jin XY, Xuan Y, et al., 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Proc 33rd Int Conf on Neural Information Processing Systems, Article 471. https://doi.org/10.5555/3454287.3454758

Marcjasz G, Uniejewski B, Weron R, 2020. Probabilistic electricity price forecasting with NARX networks: combine point or probabilistic forecasts? *Int J Forecast*, 36(2):466-479. https://doi.org/10.1016/j.ijforecast.2019.07.002

Miao KC, Han TT, Yao YQ, et al., 2020. Application of LSTM for short term fog forecasting based on meteorological elements. *Neurocomputing*, 408:285-291. https://doi.org/10.1016/j.neucom.2019.12.129

Min K, Kim D, Park J, et al., 2019. RNN-based path prediction of obstacle vehicles with deep ensemble. *IEEE Trans Veh Technol*, 68(10):10252-10256. https://doi.org/10.1109/TVT.2019.2933232

Nóbrega JP, Oliveira ALI, 2019. A sequential learning method with Kalman filter and extreme learning machine for regression and time series forecasting. *Neurocomputing*, 337:235-250. https://doi.org/10.1016/j.neucom.2019.01.070

Ran XD, Shan ZG, Fang YF, et al., 2019. An LSTM-based method with attention mechanism for travel time prediction. *Sensors*, 19(4):861. https://doi.org/10.3390/s19040861

Syriopoulos T, Tsatsaronis M, Karamanos I, 2021. Support vector machine algorithms: an application to ship price forecasting. *Comput Econ*, 57(1):55-8. https://doi.org/10.1007/s10614-020-10032-2

Vaswani A, Shazeer N, Parmar N, et al., 2017. Attention is all you need. Proc 31st Int Conf on Neural Information Processing Systems, p.6000-6010. https://doi.org/10.5555/3295222.3295349

Viccione G, Guarnaccia C, Mancini S, et al., 2020. On the use of ARIMA models for short-term water tank levels forecasting. *Water Supply*, 20(3):787-799. https://doi.org/10.2166/ws.2019.190

Wang C, Tian R, Hu J, et al., 2023. A trend graph attention network for traffic prediction. *Inform Sci*, 623:275-292. https://doi.org/10.1016/j.ins.2022.12.048

Wang ZP, Qu JF, Fang XY, et al., 2020. Prediction of early stabilization time of electrolytic capacitor based on ARIMA-Bi_LSTM hybrid model. *Neurocomputing*, 403:63-79. https://doi.org/10.1016/j.neucom.2020.03.054

Wu N, Green B, Ben X, et al., 2020. Deep transformer models for time series forecasting: the influenza prevalence case. Proc 37th Int Conf on Machine Learning, p.1-10.

Wu Z, Wu LJ, Meng Q, et al., 2021. UniDrop: a simple yet effective technique to improve transformer without extra cost. Proc Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, p.3865-3878. https://doi.org/10.18653/v1/2021.naacl-main.302

Xiao S, Yan JC, Farajtabar M, et al., 2019. Learning time series associated event sequences with recurrent point process networks. *IEEE Trans Neur Netw Learn Syst*, 30(10):3124-3136. https://doi.org/10.1109/TNNLS.2018.2889776

Xie YY, Lou YS, 2019. Hydrological time series prediction by ARIMA-SVR combined model based on wavelet transform. Proc 3rd Int Conf on Innovation in Artificial Intelligence, p.243-247. https://doi.org/10.1145/3319921.3319959

Xu DW, Wang YD, Jia LM, et al., 2017. Real-time road traffic state prediction based on ARIMA and Kalman filter. *Front Inform Technol Electron Eng*, 18(2):287-302. https://doi.org/10.1631/FITEE.1500381

Zhang D, Ling JW, Wei ZH, et al., 2018. Long-term traffic speed prediction based on multiscale spatio-temporal feature learning network. *IEEE Trans Intell Transp Syst*, 20(10):3700-3709. https://doi.org/10.1109/TITS.2018.2878068

Zhang L, Wang G, Giannakis GB, 2019. Real-time power system state estimation and forecasting via deep unrolled neural networks. *IEEE Trans Signal Process*, 67(15):4069-4077. https://doi.org/10.1109/TSP.2019.2926023

Zhang TJ, Song S, Li SG, et al., 2019. Research on gas concentration prediction models based on LSTM multidimensional time series. *Energies*, 12(1):161. https://doi.org/10.3390/en12010161

Zheng JH, Huang MF, 2020. Traffic flow forecast through time series analysis based on deep learning. *IEEE Access*, 8:82562-82570. https://doi.org/10.1109/ACCESS.2020.2990738

Zhou HY, Zhang SH, Peng JQ, et al., 2021. Informer: beyond efficient transformer for long sequence time-series forecasting. Proc 35th AAAI Conf on Artificial Intelligence, p.1-15. https://doi.org/10.1609/aaai.v35i12.17325